Treball de Fi de Grau

## Grau en Enginyeria en Tecnologies Industrials

# Aplicacions de Realitat Augmentada utilitzant el dispositiu Kinect

**Memòria**

**Autor:** Zazurca Guañabens, José Mª
**Director:** Susín Sanchez, Toni
**Convocatòria:** Gener 2020


Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

# RESUM

El món de la tecnologia i, més concretament, el de la realitat virtual és un sector que actualment no para de créixer i cada cop es desenvolupen més aplicatius que anys enrere semblaven inabastables. Tanmateix, dins d'aquesta gran expansió ens centrarem en la part més visual.

Aquest projecte consisteix en fer participar a l'usuari dins d'una escena virtual basada en realitat augmentada on serà capaç de tocar un piano virtual amb els seus peus. Per aconseguir-ho treballarem amb un software especialitzat en el desenvolupament d'escenes virtuals anomenat Unity. Al mateix temps, serà bàsic el rol d'un dispositiu anomenat Kinect ja que ens permetrà posicionar l'usuari dins de l'escena virtual.

A la memòria començarem explicant com funciona aquesta plataforma de desenvolupament que es diu Unity, tot posant èmfasis amb les dificultats que s'han hagut d'afrontar ja que era un camp totalment desconegut abans de dur a terme aquest projecte.

Posteriorment en el capítol 6 parlarem del dispositiu Kinect tot explicant com funciona la seva càmera RGB i com aquesta recull la informació per a utilizar-la després. Al mateix temps, en el capítol 7 s'explicarà com aquesta eina de treball s'implementa amb Unity per tal d'aconseguir escenes virtuals.

Seguidament en el capítol 8 ja ens centrarem en descriure pas per pas l'aplicatiu final que consisteix en una escena virtual a partir de la qual l'usuari podrà tocar un piano amb els seus peus, tot activant unes tecles virtuals que apareixen en pantalla.

I finalment, es realitzarà un pressupost del que suposa dur a terme aquest projecte juntament amb les conclusions.

# 1. SUMARI

# 2. Glossari

Totes les definicions estan adaptades al context on apareixen:

- CRV: Centre de Realitat Virtual, a la Facultat de Matemàtiques on s'ha desenvolupat tot el projecte.
- RV: Realitat Virtual.
- Escena: Lloc virtual on es fa la creació del projecte. Conforma la pantalla principal de la plataforma Unity.
- SDK: Software Development Kit (Kit de desenvolupament de software).
- Plugin: Extensió externa al software.
- Loop: Bucle.
- Script: Fitxer de text que conté el codi de programació en Unity.
- Frame: Imatge instantània. Per a cada instant que es visualitza una imatge es realitzen tots els càlculs necessaris per actualitzar l'escena.
- Joints: Conjunt d'articulacions que formen part de l'esquelet detectat pel dispositiu.
- GameObject: Element que forma part d'una escena de Unity.
- RGB: Red, Green i Blue (vermell, verd i blau en anglès). Són els colors bàsics a partir dels quals s'obtenen els demés.

# 3. Prefaci

## 3.1 Origen del projecte

Aquest projecte forma part d'un grup de projectes que s'estan desenvolupant en aquests últims anys a l'Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (ETSEIB) impulsats pel professor i director d'aquest projecte Toni Susín. Aquest treball s'ha desenvolupat al Centre de Realitat Virtual de la facultat de Matemàtiques i Estadística (FME), a partir del qual s'han pogut obtenir els dispositius d'immersió en Realitat Virtual.

Seguint amb els treballs que s'havien realitzat anys anteriors, es proposa expandir l'aprenentatge de la plataforma Unity i dels dispositius que disposa aquesta tecnologia per aconseguir una immersió de l'usuari cada cop més realista dins una aplicació de la realitat virtual. En aquest projecte en concret només ens hem basat en el dispositiu Kinect V2, amb l'objectiu d'aprendre com capta informació, analitza i la utilitza posteriorment per aconseguir millorar aquesta immersió.

Aquest projecte precisament sorgeix d'un projecte molt semblant que ja havia guiat el professor Toni Susin i que es volia dur a terme un altre cop tot aplicant certes millores. Cal destacar que al llarg del projecte han sigut de gran ajuda els projectes d'anys enrere per resoldre moments de dubte que han anat sorgint.

## 3.2 Motivació

Aquest va ser un projecte que va sorgir d'un interès recíproc entre jo i el meu tutor Toni Susín.

Per una banda, personalment sempre m'ha interessat tot el que vingui relacionat del món de les tecnologies, dins del qual em sorprenen els constants avenços en l'àmbit de la Realitat Virtual que podem veure per exemple des de ben aprop cada any en el Mobile World Congress que s'organitza aquí a Barcelona. A més, al informar-me de la plataforma Unity vaig descobrir que s'havien creat múltiples videojocs populars, pel que seria molt interessant treballar-hi. A partir d'aquí, vaig afrontar aquest projecte com una oportunitat per endinsar-me

dins d'aquest àmbit de la Realitat Virtual que era totalment desconegut per a mi, exceptuant petits coneixements de programació adquirits durant la carrera. Tanmateix, no puc amagar el respecte que em genera dur a terme un projecte d'un àmbit nou i haver-lo d'afrontar des de zero. Per altra banda, vaig tenir la sort de que el professor Toni Susín estigués interessat en guiar més projectes d'aquest camp.

## 3.3 Requeriments previs

Per a desenvolupar aquest projecte únicament s'han requerit coneixements de programació adquirits en vàries assignatures del grau en Enginyeria en Tecnologies Industrials. Tanmateix, cal destacar que en aquest cas es treballaria amb un llenguatge totalment nou anomenat C# i que, per conseqüència, és important haver adquirit els aspectes bàsics del llenguatge de programació treballat a la carrera per agilitzar el procés d'aprenentatge.

A partir d'aquí, cal ser conscient que al endinsar-se en el món de la programació i especialment amb un llenguatge nou podran sorgir moments de bloqueig que caldrà afrontar-los amb paciència. Per sort, el motor de jocs amb el treballarem, Unity, proporciona una plataforma de recerca on trobarem explicacions i exemples de totes les comandes de programació en cas de dubte.

# 4. INTRODUCCIÓ

## 4.1 Objectius del projecte

La realitat virtual és un conjunt d'escenes i objectes aparentment reals. Per aconseguir-ho disposem de diversos softwares de simulació que permeten la seva simulació a partir de certs dispositius com unes ulleres o cascs de realitat virtual, o en el nostre cas a partir de la Kinect V2. El més important d'aquesta simulació és aconseguir la immersió de l'usuari dins d'una escena virtual.

En altres projectes semblants s'ha aconseguit aquesta immersió a partir de convertir l'usuari en un avatar que reprodueixi els seus moviments amb la màxima precisió. Altres aplicacions s'han basat en que l'usuari fos simplement un espectador de l'escena virtual, fet que dificulta bastant aconseguir aquest objectiu d'immersió.

Si més no, tal i com s'ha avançat anteriorment, en aquest projecte treballarem amb el dispositiu Kinect V2 que visualitzarà una escena real on ens trobarem immersos i a partir d'aquí afegirem objectes virtuals amb els quals l'usuari pugui interactuar. L'objectiu d'aquest projecte serà aconseguir que aquesta interacció sigui el màxim de precisa, fins arribar al punt que doni la sensació que aquell objecte forma part del món real. Per això, caldrà abans de tot conèixer com funciona la tecnologia d'HTC i com treballar amb les dades que aquesta ens proporciona.

## 4.2 Abast del projecte

El projecte consta de diferents etapes que s'han dut a terme per assolir els objectius exposats anteriorment. Abans de tot, al tractar amb dispositius totalment desconeguts, s'ha realitzat un període d'introducció que ha servit com aprenentatge del motor de jocs Unity i de la Kinect V2 per estudiar la seva tecnologia i treure-li el màxim de profit.

Seguidament ja s'ha començat a gestionar l'escena virtual del piano per aconseguir una immersió molt precisa. S'ha intentat abordar detalls com l'altura de la Kinect V2 per aconseguir que aquesta no dificultés la simulació del piano virtual segons la seva disposició.

Tanmateix, al treballar amb el dispositiu Kinect V2 ens hem hagut d'adaptar a les seves limitacions que s'explicaran més endavant.

# 5. UNITY: LA PLATAFORMA DE DESENVOLUPAMENT

## 5.1 Introducció a la plataforma

Unity és un motor de videojocs multiplataforma creat per l'empresa Unity Technologies, creada a Dinamarca l'any 2004. L'objectiu d'aquesta plataforma era ajudar a tots aquells desenvolupadors independents que no tinguessin recursos suficients per crear el seu propi motor de joc i que, d'aquesta forma, poguessin dur a terme els seus projectes.

Actualment Unity destaca per ser un motor de jocs econòmic i accessible per a tot el públic. També cal destacar que presenta una interfície gràfica molt simple i intuïtiva i que, a més a més, té una estructura que permet minimitzar molt el codi de programació per realitzar diverses accions. Aquest últim aspecte facilita molt l'aprenentatge per part dels usuaris que poden tirar-se enrere degut al seu desconeixement en aspectes més tècnics com la programació.

Finalment Unity permet que un cop s'hagi creat l'aplicació desitjada, aquesta pugui ser compilada en diferents aplicacions com Windows, iOS, Linux o Android passant per altres plataformes com Xbox.

## 5.2 El llenguatge de programació: C#

El llenguatge que s'utilitza per programar en Unity és el C# (pronunciat C sharp en anglès). És un llenguatge que deriva del C i C++ que va ser desenvolupat i estandarditzat per Microsoft com a part de la seva plataforma .NET.

A llarg d'aquests mesos al Centre de Realitat Virtual (CRV) s'ha utilitzat el Visual Studio 2015 com a entorn de desenvolupament integrat (IDE) del projecte. A partir d'aquest

programa s'han creat tots els scripts necessaris, en la majoria dels casos sense gaires dificultats gràcies a la guia que podem trobar a Unity, tal i com s'ha comentat anteriorment, on tenim totes les indicacions per utilitzar adientment les comandes que vulguem. Al mateix temps, cal destacar la fàcil detecció d'errors que presenta aquest programa ja que en tot moments ens indica en quina línia de codi tenim l'error.
El C# és un llenguatge molt elaborat on l'usuari pot trobar moltes formes de fer una mateixa acció i la dificultat es troba en escollir el camí més ràpid i eficient. Tanmateix, l'aprenentatge d'aquest nou llenguatge no ha sigut molt complex gràcies als coneixements adquirits durant el grau en les assignatures d'Informàtica on treballem amb el Python i, a més a més, a nivell personal he tingut l'avantatge que a la meva escola ja havia treballat amb el C++ anteriorment. Per tot això, es pot afirmar que no ha sigut un gran impacte la necessitat de treballar amb un programa com el C#.

## 5.3 Funcionament bàsic de Unity

Tal i com s'ha vingut comentant en apartats anteriors, Unity destaca per ser un programa molt simple i intuïtiu. Per funcionar sempre s'ha de crear un projecte, el qual pot estar compost per diverses escenes en les que a cada una podem anar situant els elements protagonistes anomenats "GameObjects". Aquests elements tenen múltiples formes, des d'un cub 3D fins a un text, o fins i tot hi ha han cops que ens ajuda crear GameObjects buits, els quals no tenen cap representació visual a l'escena. Un cop creats aquests elements, podem modificar el seu comportament a partir d'assignar-los per exemple un color, una condició física com la gravetat o directament associar-los un Script o, en altres paraules, un fitxer de text que presenta un codi que estableix el seu comportament.

Al llarg de la creació d'una escena podem afegir múltiples elements, però sempre ha de constar de dos objectes que ja venen establerts inicialment. El primer és la càmera que s'encarrega de filmar el que passa a l'escena un cop apretem el botó de 'play'. Aquesta càmera la podem situar on vulguem de l'escena o fins i tot podem rotar-la per aconseguir un punt de vista concret. I el segon element indispensable és la llum, en aquest cas, una llum direccional que representa el sol i que il·lumina l'escena.

A continuació veurem una imatge on tindrem les 5 finestres amb les que es treballa a l'hora de crear una escena de Unity i les anirem explicant per separat.

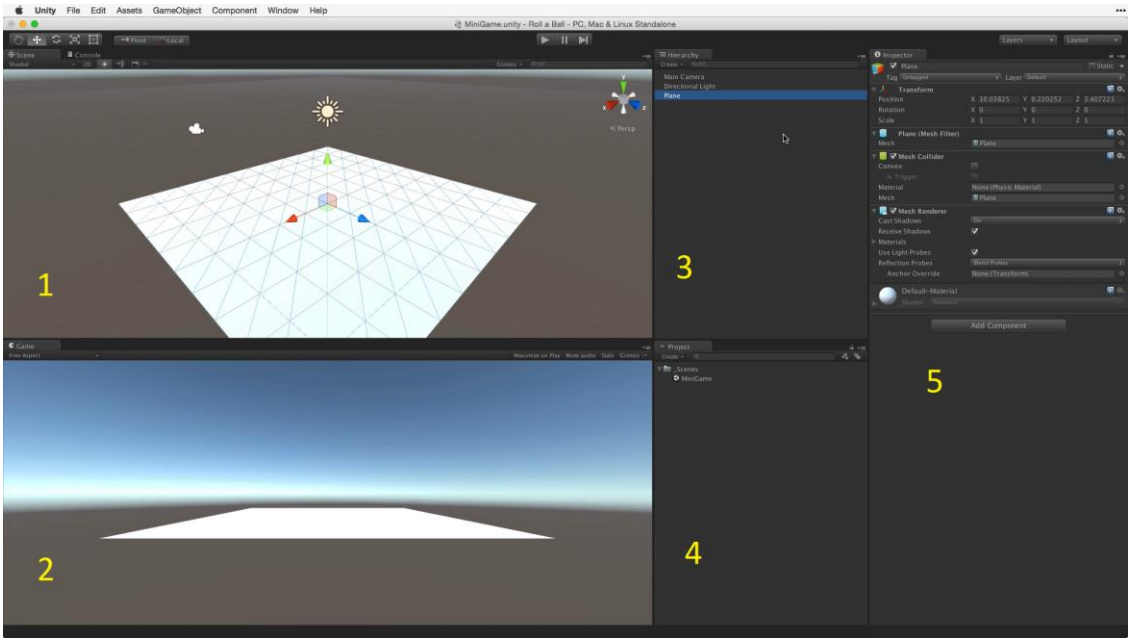



*Figura 1. Interfície gràfica d'usuari del programa Unity*

En primer lloc la número 1 és la finestra anomenada 'Scene View'. Aquesta és la finestra que considerem més general ja que és on seleccionarem i situarem els entorns, la càmera, el jugador i tota la resta de GameObjects. Serà important controlar les dimensions de l'espai per situar tots aquests elements correctament.

En segon lloc la número 2 és la 'GameView'. Tal i com diu el seu nom, aquesta finestra serà la visió que tindrem de l'escena un cop apretem el botó del play, en altres paraules, és la representació del joc un cop ja finalitzat.

La tercera finestra es tracta de la 'Hierarchy Window'. En aquesta es visualitzen tots els GameObjects que es van ubicant dins de l'escena per ordre de creació, tot i que els podem reordenar manualment arrastrant-los cap amunt o cap abaix. Dins d'aquesta llista molt freqüentment trobarem el concepte de 'jerarquia' que es basa en GameObjects que es troben dins d'un altre que seria el 'pare' en clau familiar. Com a conseqüència, qualsevol modificació aplicada al 'pare' és aplicada automàticament als 'fills'. Aquest concepte s'explicarà àmpliament en apartats futurs. Finalment, si algun GameObject és eliminat de l'escena també s'elimina d'aquesta llista. En l'exemple de la imatge observem que tenim els 2 elements explicats anteriorment, càmera i llum, i un pla que actua com a terra de l'escena.

La quarta finestra és la 'Project Window', on podem accedir i gestionar tots els assets que pertanyen a l'escena. A la part esquerra d'esquerra tota la llista dels assets en forma de jerarquia. Durant la creació d'una escena podem arribar a utilitzar múltiples arxius i per això és important tenir aquesta finestra ben ordenada.

Finalment tenim la cinquena finestra anomenada 'Inspector'. En aquesta podem veure i modificar les característiques de cada GameObject de l'escena, com seria la seva posició, les seves dimensions o el seu Script associat en cas de necessitar-ho.

## 5.4 Roll-a-ball

Tal i com s'ha vingut explicant, la gran dificultat amb la que ens afrontem en aquest projecte és el desconeixement inicial del programa Unity. Per això, abans de començar a dur a terme el nostre propi projecte era imprescindible familiaritzar-se amb el programa mitjançant algun exemple senzill. Això es va aconseguir a partir de seguir els passos d'un tutorial que podem trobar a la web de Unity anomenat 'Roll-a-ball', el qual venia recomanat pel meu tutor.

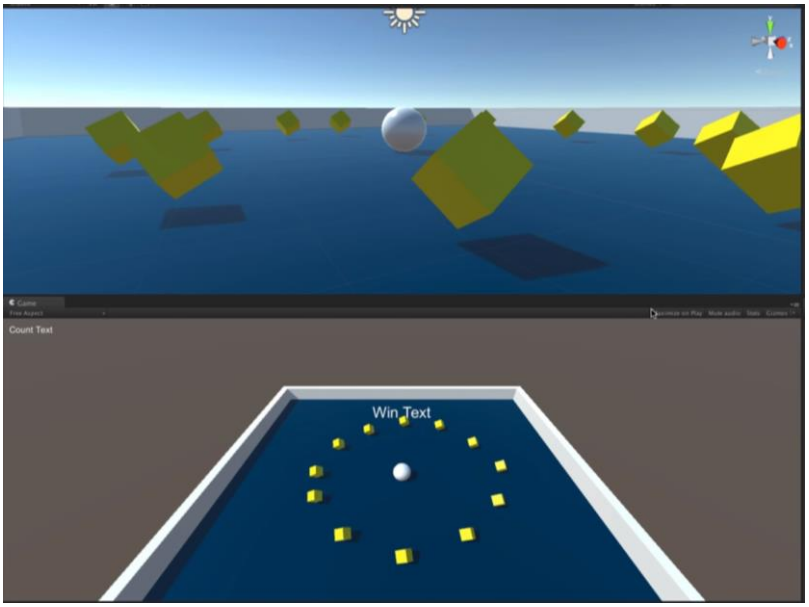



*Figura 2. Captura de pantalla del mini joc Roll-A-Ball*

Aquest joc consisteix en dirigir una bola mitjançant el teclat amb l'objectiu de recollir els 12 cubs que es troben en un recinte quadrat i que aniran desapareixent a mesura que siguin recollits. Aquest recinte presenta unes parets per tal que no caigui la bola al buit i, al mateix temps, s'afegeix un comptador de cubs adalt a l'esquerra del GameView.

Un cop finalitzat el tutorial proposat, es va voler aprofundir en el joc i es van afegir 2 opcions per complicar una mica més el joc. Per una banda es va voler que al recollir cada cub aquest es transformés en una explosió visual com a simulació del xoc entre la bola i el cub. I per altra banda també es va imposar que al produir-se cada xoc de la bola amb cada un dels 12 cubs es reproduís el so d'una explosió. Aquest últim apartat cal destacar que era de gran ajuda de cara al meu projecte, on hauríem de simular sons de notes musicals.

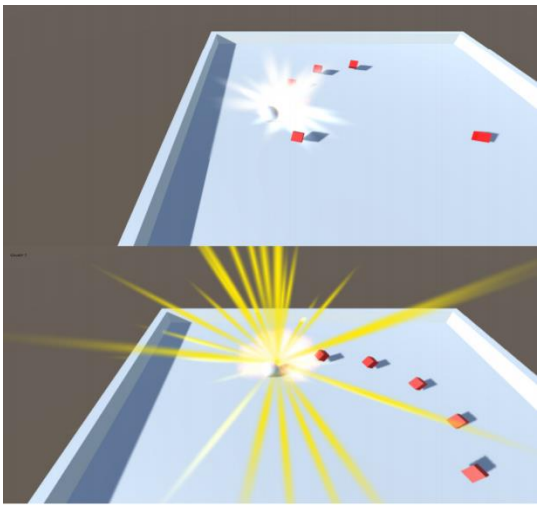


*Figura 3. Captura de pantalla del mini joc Roll-A-Ball amb explosions*

Finalment, després d'aquest tutorial i les seves petites complicacions, ja es va considerar que s'havien assolit els coneixements suficients per afrontar el projecte proposat.

# 6. EL DISPOSITIU KINECT

## 6.1 Història i versions del dispositiu

Ens hem de remuntar fins el 4 de novembre de l'any 2010 als Estats Units per trobar la primera versió del dispositiu Kinect, anomenat Kinect, que va ser llançada al mercat per l'empresa Microsoft per tal de millorar l'experiència amb la videoconsola Xbox 360. L'objectiu d'aquesta eina era aconseguir una millor sensació d'immersió que la que l'usuari podia obtenir a partir dels anteriors comandaments convencionals. Més concretament, es tractava d'una nova interfície d'usuari on es puguessin reconèixer els gestos i ordres de veu sense necessitat de tenir contactes físics amb els comandaments.


*Figura 4. Fotografia de la primera versió de la Kinect*

Posteriorment, el 22 de novembre de 2013 ja va sortir al mercat el nou model que serà el que s'utilitzarà en aquest projecte, que porta el nom de Kinect V2 i que presenta diverses millores respecte la primera versió comentada.

*Figura 5. Fotografia de la Kinect V2*

## 6.2 Com funciona la Kinect V2

La kinect V2 és un dispositiu de processament digital que funciona com una càmera 3D ja que, a part de la gravació d'imatges, és capaç de reconèixer i mesurar la distància a la que es troben els objectes del seu voltant. La seva sensibilitat es tal de detallada que és capaç de diferenciar si la persona que s'està gravant està obrint la boca o bé somrient. A més a més, cal destacar que aquesta mesura de profunditat és totalment independent de la llum exterior gràcies a treballar amb llums infrarojos. Aquest detall pot ser de gran utilitat pels jugadors de videojocs que prefereixen jugar a les fosques.
En total aquest dispositiu consta de 4 parts: una càmera RGB, un emissor d'infrarojos (Active IR), un sensor de profunditat IR i un micròfon de múltiples matrius.

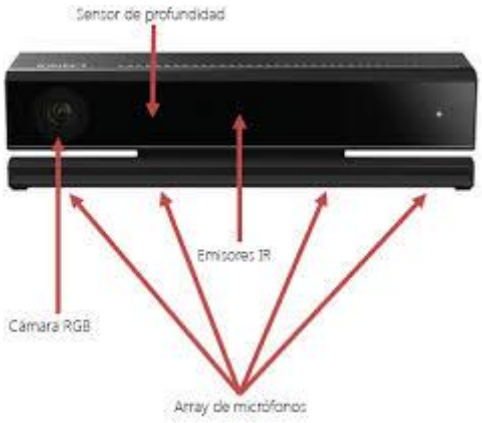



*Figura 6. Fotografia de la Kinect V2 amb cada una de les seves parts*

I com aconsegueix la Kinect V2 mesurar la profunditat del objectes en gravació? Doncs això és possible gràcies a la tecnologia anomenada ToF (Time of Flight). El funcionament d'aquesta es basa en l'emissió de senyals de llum infraroja i en la mesura del temps que tarden aquests en retornar, calculant així la distància a la que es troba cada píxel. Finalment a partir de la mesura de tots els píxels es pot reconstruir la imatge en profunditat.
Cal destacar que la llum emesa per part de la Kinect V2 es realitza de forma indirecta. Aquesta manera consisteix en la modulació d'una ona continua sinusoidal que es reflecteix sobre els objectes i, per conseqüència aquesta queda desfasada, tal i com veient en la següent imatge:

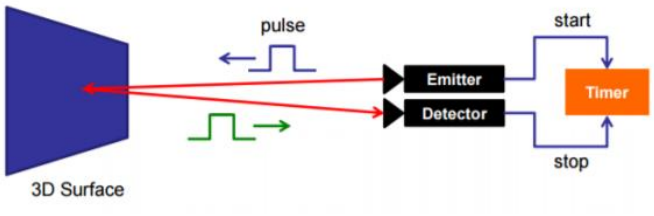



*Figura 7. Esquema del funcionament de la tecnologia ToF*

A partir d'aquest desfasament el dispositiu és capaç de calcular la distància de cada píxel, mitjançant la fórmula que veurem a continuació on *d* és la distància de l'objecte, *c* la velocitat de la llum, *w* la freqüència de modulació i *phi* el desfasament entre la fase inicial i final.

$$d = \frac{c}{4\pi\omega}\phi$$

(Eq.1)

## 6.3 Informació proporcionada

### Càmera RGB

Tal i com s'ha comentat anteriorment, la Kinect V2 treballa amb una càmera RGB que concretament presenta una resolució 1920x1080 (1080p) amb un camp de visió de 70º verticals i 60º horitzontals. La gravació d'imatges es realitza a uns 30 fps (frames per second) aproximadament. Aquesta càmera ha estat fonamental per aquest projecte ja que és l'encarregada de visualitzar l'usuari tocant les tecles del piano virtual.

### Sensor de profunditat

Aquest sensor presenta un rang de profunditat de 0,4 fins els 4,5 metres i presenta una resolució de 512x424 píxels. Mitjançant una escala de grisos podem diferenciar els objectes més propers i els més distants respecte el dispositiu.

### Micròfons

A més a més, la Kinect V2 aporta un micròfon per aconseguir el reconeixement per veu. Tanmateix, aquest no ha estat utilitzat en tot el projecte ja que la seva informació no era rellevant.

### Seguiment de l'esquelet

L'aspecte més rellevant de la Kinect V2 i que ha resultat imprescindible en la creació d'aquest projecte és poder realitzar el seguiment de l'esquelet que graba el dispositiu. En total es poden detectar fins a 6 esquelets de persones, els quals estan formats per la unió de 25 joints (articulacions) del cos simulant la unió dels ossos principals dels usuaris.
A continuació veurem una imatge on es mostra tal i com el dispositiu reprodueix l'esquelet d'una persona.

*Figura 8. Seguiment de l'esquelet*

Per aconseguir una precisa orientació dels cossos segons la posició del cos la Kinect V2 realitza la detecció dels joints en 3D.

Més concretament, això s'aconsegueix gràcies a un algoritme creat per Microsoft en el qual la clau principal és el processament de la imatge en profunditat. A partir d'ella, el dispositiu és capaç de classificar cada píxel detectar a l'hora de rastrejar el cos que es troba davant la pantalla. A continuació veurem una imatge on es mostra aquest procediment.

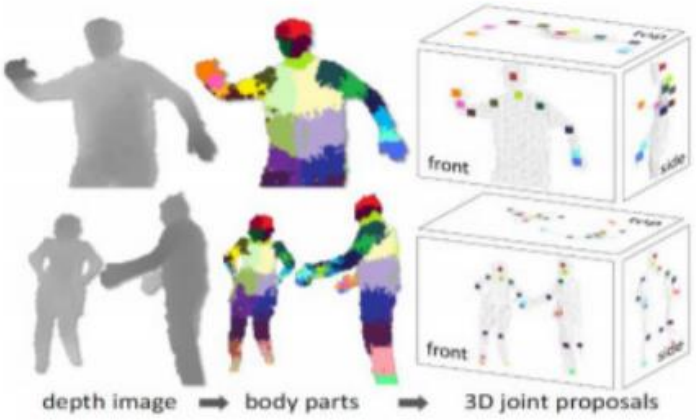



*Figura 9. Funcionament de detecció del seguiment d'un esquelet*

## 6.4 Limitacions del dispositiu

La kinect V2 és un dispositiu molt útil amb unes característiques molt aprofitables. Si més no, a l'hora de començar el projecte també s'han de tenir en compte les seves limitacions que poden dificultar diverses situacions:

- ➢ No diferenciació entre davant i darrera
    - La kinect V2 és un dispositiu preparat per gravar i detectar les persones de manera frontal ja que se suposa que sempre es trobaran de cara la pantalla. Per això, no és capaç de distingir si l'usuari es troba de cara o d'esquenes. s.
- ➢ No detecció dels elements que no són visibles
    - A l'hora de posicionar i orientar les articulacions del cos que es té davant, la Kinect es basa en el sensor de profunditat. Si més no, si alguna part del cos no es troba davant de la càmera, com per exemple un braç darrera l'esquena, aquest no pot ser detectar pel dispositiu.
- ➢ Complexitat per la representació de les mans
    - Per últim cal destacar que tot i que les mans estan representades per 3 joints cada una, la seva gran sensibilitat al moviment és tal que el dispositiu no pot dur a terme un seguiment adient.

Sortosament, cap d'aquestes tres limitacions comentades ens afecta en la realització d'aquest projecte ja que només ens interessa que la Kinect gravés correctament els 2 peus de l'usuari que tindrem en front.

# 7. Implementació del Kinect amb Unity

## 7.1 Introducció: aplicacions del dispositiu i antecedents

Com s'ha anat explicant, la Kinect V2 destaca per ser un dispositiu que permet detectar la profunditat dels objectes o persones que tingui davant seu, fet que en el últims anys ha ajudat a molts sectors de la tecnologia a evolucionar.

Per començar, un exemple[1] seria el camp dels llenguatges dels signes on uns investigadors a la Xina van desenvolupar un sistema que gràcies a la Kinect es captaven els gestos, s'interpretaven i finalment es traduïen de forma oral o escrita. De la mateixa forma, també es pot realitzar el procés a l'inversa a partir d'un avatar que a partir del que li diguem ho transforma en els gestos adequats.

Un altre exemple seria en el camp de la medicina, on gràcies a la Kinect V2 els pacients poden recuperar-se des de casa a partir de diversos exercicis. Aquests poden ser monotoritzats i regulats pel sistema i és que els investigadors creuen que l'atmosfera del joc pot accelerar la recuperació de la persona.

I per descomptat en el món de l'animació i els videojocs és evident la gran utilitzat que representa aquest dispositiu ja que a partir d'ell podem detectar moviments de persones i aplicar-los en models 3D. De fet, aquest projecte no seria possible sense utilitzar aquesta detecció de les persones, ja que necessitem saber concretament on l'usuari té els peus per tal de captar els seus moviments a l'hora de tocar el piano virtual.

## 7.2 Plugins per Unity

Unity és un programa que ha evolucionat més del que els seus propis creadors esperaven. I és que principalment Microsoft només contemplava utilitzar-lo com a controlador de jocs. Tanmateix, amb la versió PC, l'empresa va veure's obligada a expandir-lo i per això va crear una llibreria anomenada SDK (Software Development Kit) que permet la interacció entre el motor de jocs i l'ordinador. Al mateix temps, per tal d'accedir al màxim número de plataformes Microsoft va crear un "plugin" per Unity, que vindria a fer la funció de pont entre

[1] https://www.hongkiat.com/blog/innovative-uses-kinect/

la Kinect i Unity. Aquest plugin es pot descarregar fàcilment des de l'ordinador i consta de 3 arxius i 2 escenes d'exemples. Els arxius s'anomenen Kinect 2.0, KinectFace 2.0 i Kinect.VisualGestureBuilder.2.0. mentres que les escenes es diuen GreenScreen i KinectView. En la següent imatge veiem una captura de pantalla dels elements esmentats al descarregar el plugin:

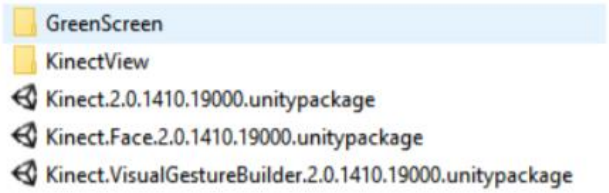



*Figura 10. Arxius del plugin*

Un cop descarregat, només cal obrir Unity i importar-lo en el projecte com a "Custom Package". Per conseqüència, tots els arxius necessaris se'ns apareixen en la finestra del projecte, a partir dels quals serem capaços d'accedir a la informació que ens proporciona el dispositiu.

Cal recordar que davant de qualsevol dubte que se'ns presenti durant el projecte sobre el motor de jocs Unity tenim una xarxa de fòrums gegant amb milers de persones al darrera. Concretament aquestes persones intervenen sovint en els seus fòrums oficials i, fins i tot, existeixen professionals que la seva feina consisteix en proposar solucions i donar consells a aquells usuaris que estan desenvolupant aplicacions. Podem denominar aquestes plataformes com un "win to win", és a dir, que les dues parts en surten beneficiades. Per una banda, els usuaris no tenen por a desenvolupar projectes ja que saben que tindran professionals darrera per ajudar-los i per altra banda, les plataformes de desenvolupament seran cada cop més modernes ja que hi hauran molts usuaris interessats en participar-hi. Tot això es pot afirmar en primera persona ja que en aquest projecte durant moments de bloqueig aquests fòrums han estat de gran ajuda.

Al descarregar el plugin complet aquest incorpora una escena anomenada KinectView. Aquesta serveix per familiaritzar-se amb les diferents fonts d'informació que proporciona la Kinect com són les de profunditat, color o seguiment de l'esquelet.

Si més no, abans d'explicar les que ens han servit per aquest projecte, cal explicar quins passos segueix el dispositiu a l'hora de gestionar el flux d'informació de cada font.

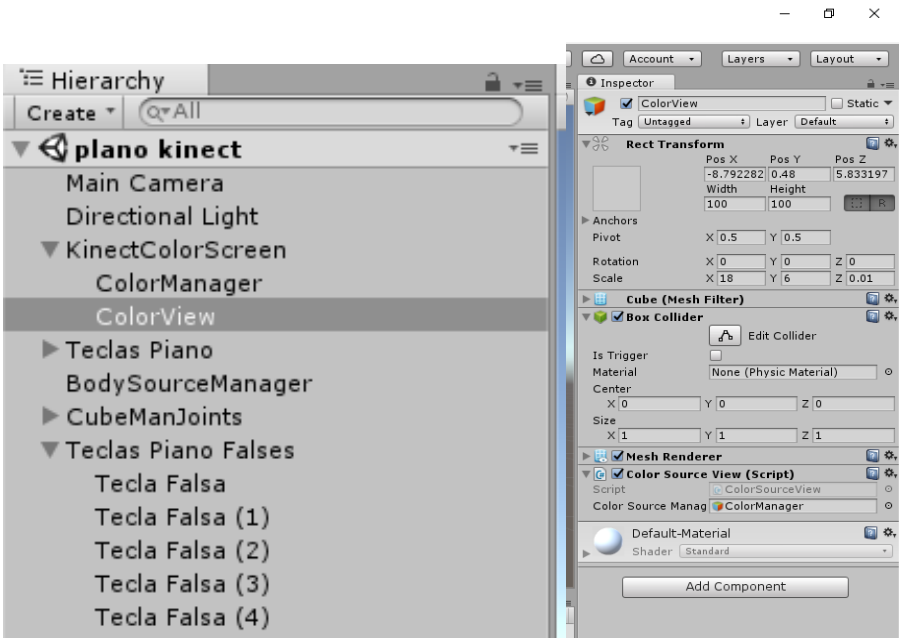

*Figura 11. Flux de programació de Kinect V2*

En la imatge de sobre tenim un breu resum d'aquest procés. Tal i com veiem, es comença per identificar el sensor i seguidament s'accedeix a la font ("Source") de la qual l'usuari vol informació. Cada font presenta un lector ("Reader") associat, i és aquest últim el que accedeix a la informació de cada frame, la qual es representa d'una manera o d'una altra depenent del tipus de font.

Per aquest projecte només s'ha treballat amb la *ColorFrameSource*, la qual reprodueix cada píxel en color de la càmera, i amb la BodySource que ens permet representar el seguiment de l'esquelet a través del punts pels joint i línies verdes pels ossos. A partir d'aquí, per cada font amb la que treballem sempre tindrem dos scripts representats a l'escena: SourceManager i Source View. En el nostre cas al treballar amb la font ColorFrameSource, el primer (ColorSourceManager) és l'encarregat de captar la informació de la càmera RGB, la qual és representada seguidament per pantalla pel segon script (ColorSourceView).

Un cop tenim aquests 2 scripts és important entendre com aquests es combinen per tal de reproduir per pantalla allò que graba la Kinect, i és que aquest procés és utilitzat amb freqüència al llarg d'aquest projecte. En primer lloc, hem de crear un GameObject que l' anomenem ColorManager ja que a continuació li assignem el script del ColorSourceManager. Seguidament, crearem un altre GameObject però aquest cop de tipus "Cube", proporcionant-li unes dimensions com una pantalla, és a dir, gran longitud i amplada però molt poca profunditat. A aquest element li direm ColorView i li assignem el codi del ColorView per visualitzar la informació. Finalment, i aquí ve el pas diferencial per relacionar els 2 arxius, es crea una variable pública dins del script ColorView. D'aquesta manera posteriorment ens apareix a l'inspector aquesta variable i serà allà on arrastrem el GameObject buit creat inicialment.

Cal destacar que per entendre ràpidament aquest procés han sigut de gran ajuda els anteriors projectes semblants que em va proporcionar el meu tutor. En la següent imatge podem veure com queden finalment les pantalles de Unity.




*Figures 12 i 13. Disposició de Unity un cop afegit el plugin*

Tanmateix, podem observar com la part essencial per a que tot això funcioni és programar correctament els scripts corresponents. Per entendre'ls s'explicarà d'exemple el codi de BodySourceManager, el qual gestiona les dades de l'esquelet:

1. S'importen les biblioteques de la Kinect.
2. Es genera una variable per emmagatzemar les dades del sensor, una segona pel lector i finalment una tercera per les dades del cos.
3. S'executa la funció Start () només un cop per tal d'iniciar el joc, obrir el sensor i crear un lector.
4. La funció Update () és l'encarregada d'anar llegint les dades de l'últim frame i anar-lo actualitzant. Més concretament, s'encarrega de guardar la informació de l'esquelet visualitzat i emmagatzemar la informació dels 25 joints que formen cadascun.
5. La funció OnApplicationQuit () s'encarrega de tancar el sensor i el lector un cop es surt de l'aplicació.

A continuació mostrarem el codi complert on podem veure les diferents parts que s'han explicat:

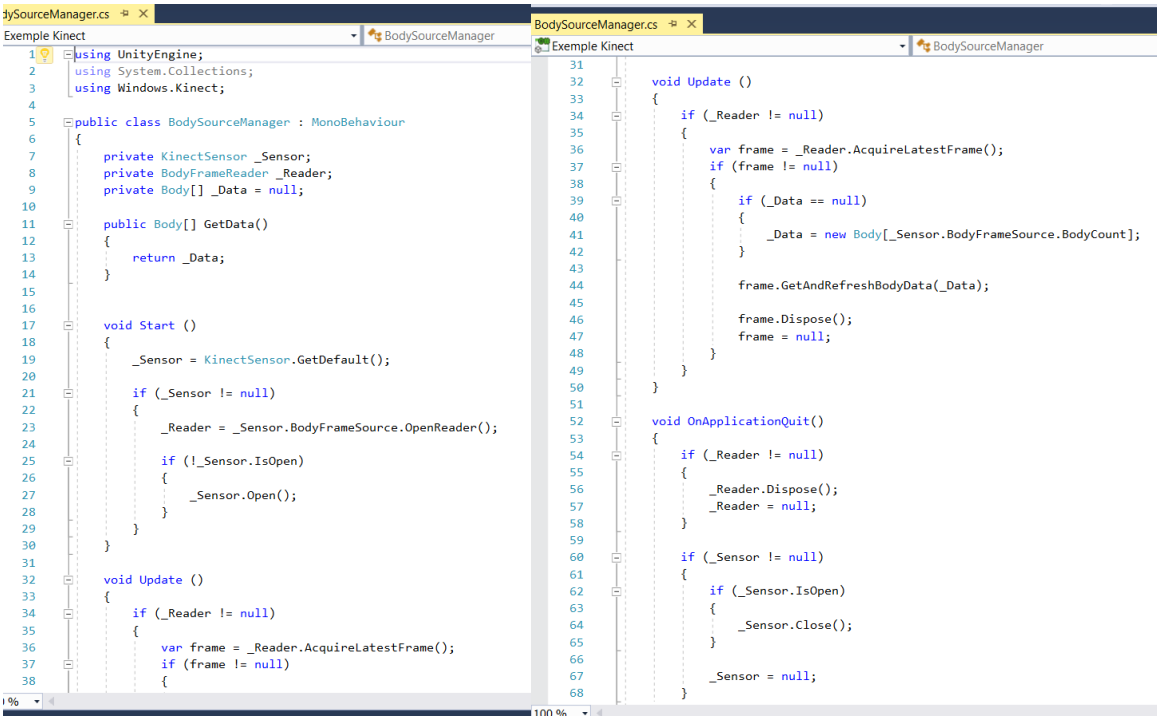



*Figura 14. Script BodySourceManager*

## 7.3 L'esquelet vist des del punt de vista de la Kinect V2

Tal i com s'ha explicat anteriorment, el dispositiu Kinect ens permet captar fins a 6 persones i representar els seus esquelets. Ara bé, com són visualitzats aquests esquelets? El que fa el dispositiu es subdividir-los en un total de 25 articulacions ("joints") que porten el seu propi nom segons la part del cos que representen. A més, en aquelles parts que són simètriques se'ls hi afegeix si representen la part dreta o esquerra com per exemple els peus que seran els joints fonamentals per aquest projecte. En la següent imatge podem veure com es representen els 25 joints al llarg de tot l'esquelet.

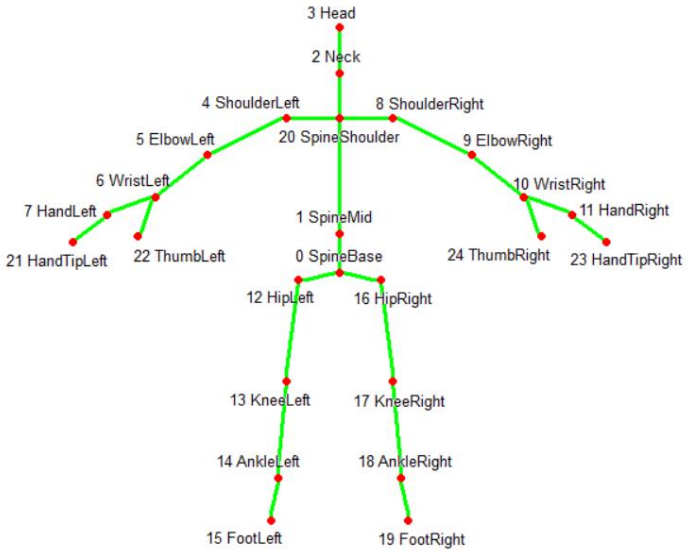



*Figura 15. Nomenclatura i jerarquia del joints de la Kinect*

Per altra banda, abans de començar a treballar amb el projecte, cal entendre el concepte d'emparentar objectes. Un GameObject pot establir-se com a 'pare' i no dependre de ningú, o bé com a 'fill' on llavors passaria a dependre d'un altre GameObject. A partir d'aquí, a mesura que es van afegint GameObjects es va establint una relació jeràrquica entre ells on qualsevol canvi (moviment, rotació eixos X,Y i Z) aplicat a un GameObject pare serà automàticament aplicat també a tots els seus fills. En el cas de la representació de l'esquelet els joints funcionen igual on el joint *SpineMid* és el pare principal i seguidament cada part del cos s'anirà ramificant fins arribar a les seves extremitats. Per això, tot i que en el meu projecte només necessitava els 2 joints que representen els 2 peus, al ser aquests fills de molts altres joints vaig haver d'introduir-los tots.

## 7.4 Els quaternions i el "Gimbal Lock"

Avui en dia quan parlem de rotacions el mètode més utilitzat per representar-los són els anomenats Angles d'Euler. Aquests constitueixen un conjunt de tres coordenades angulars que serveixen per especificar l'orientació d'un sistema de referència d'eixos ortogonals mòbils respecte un altre sistema de referència ortogonal fixe. El fet de que la referència sigui ortogonal simplifica la comprensió d'una orientació, ja que qualsevol gir es pot descomposar mitjançant tres eixos anomenats guinyada, capcineig i balanceig que podem visualitzar millor en la següent representació a través d'un avió:

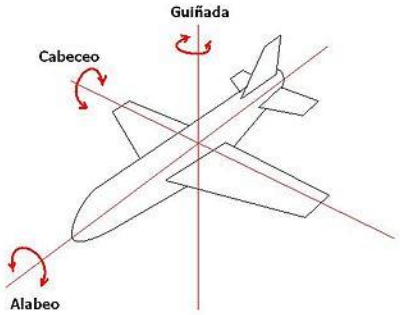



*Figura 16. Angles d'Euler d'un avió*

Una rotació es defineix com un canvi de rotació respecte un eix al llarg d'un temps determinat. En el cas tridimensional utilitzem els angles d'Euler per descomposar aquesta rotació en 3, una per cada eix. S'ha de tenir en compte que la posició final de l'element varia en funció de l'ordre en el qual es realitzen les 3 rotacions. Tanmateix, els angles d'Euler presenten un problema anomenat 'Gimbal Lock'.

Una forma de representar les 3 rotacions dels angles d'Euler són cercles de colors diferents anomenats "gimbals". En aquesta representació es pot assignar una jerarquia com en els joints de l'esquelet, on tenim un dels 3 eixos que és el 'pare', de forma que si aquest gira els altres 2 també ho fan, i al mateix temps hi ha un segon eix de jerarquia que afecta només a l'últim eix.

El 'Gimbal Lock' apareix quan dos "gimbals" coincideixen esdevenint concèntrics i, per conseqüència, es perd un grau de llibertat. Aquesta singularitat la podem observar en la següent imatge, concretament a la que trobem a la dreta de tot.

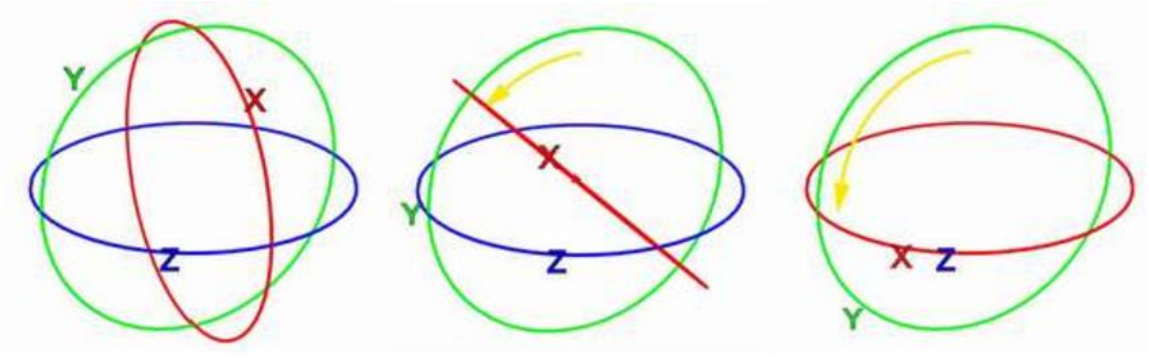



*Figura 17. Exemple de situació del Gimbal Lock*

Per tal d'evitar aquest problema Unity, Kinect i alguns softwares d'animació proposen treballar amb els anomenats quaternions. A l'hora de representar rotacions amb aquests s'utilitza un vector de quatre dimensions a partir de l'Eq.1:

$$Q = (cos(\theta/2), u * sin(\theta/2)) \qquad \text{Eq.1.}$$

A l'hora de comparar els angles d'Euler amb els quaternions observem que la major diferència és que aquests últims, en comptes de treballar amb 3 eixos, treballen amb un sol eix anomenat **u**. Tal i com veiem en la fórmula, cada una de les seves components és multiplicada pel sinus de la meitat d'un angle. Aquest angle és els graus de gir respecte el vector director **u**. Per altra banda, la primera component del vector observem que és el cosinus de la meitat d'angle, el que acaba provocant que es representi una rotació tridimensional a partir d'un vector de 4 components. A la següent figura 17 podem veure el vector u que seria el vermell i el punt w que aniria rotant al voltant del cercle verd que representa l'angle de gir.

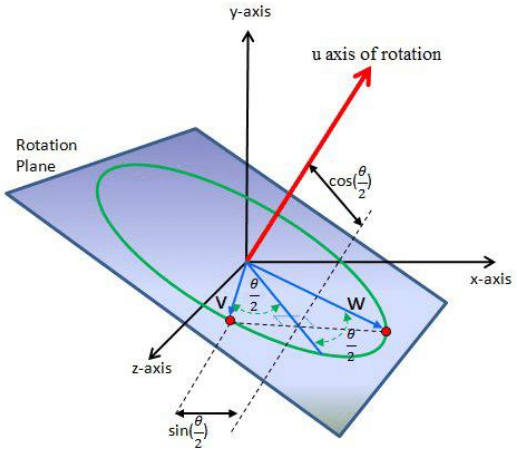



*Figura 18. Rotació amb quaternions*

Cal dir com a desavantatge d'aquesta resprentació que és poc intuitiva ja que és complicat imaginar un vector en un espai de tres dimensions per obtenir la rotació desitjada.

A l'hora d'aplicar-ho en els dispositius d'aquest projecte, per una banda la Kinect utilitza la classe interna Vector4 per a la representació dels quaternions, mentres que per altra banda Unity presenta directament una classe anomenada Quaternion. En els dos casos les classes estan constituïdes per un vector de quatre components anomenades X,Y,Z i W que venen a ser les quatre components que hem explicat anteriorment.

# 8. L'escena principal

## 8.1 Disseny de la pantalla amb les tecles de piano

La primera part d'aquest projecte es basa en el disseny de la pantalla amb les tecles de piano o, en altres paraules, en allò que realment està veient l'usuari quan interactua amb l'escena virtual.

En primer lloc, es van crear les diferents tecles que tindria el nostre piano virtual, el qual va consistir simplement en crear diversos 'GameObjects' de tipus 'Cube', donar-lis la forma rectangular que presenta una tecla i finalment pintar-les de diferents color per tal de facilitar a l'usuari la seva visualització ja que cadascuna reproduirà una nota diferent. A continuació veurem la imatge de les tecles vistes des d'amunt.

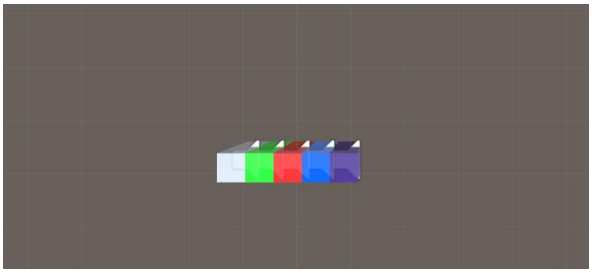


*Figura 19. Tecles del piano vistes des d'amunt*

En segon lloc, es va dissenyar la 'KinectColorScreen' que jugaria el paper de pantalla on es reproduïria tot allò que graba la Kinect. Per tal de dur-ho a terme s'han seguit els diferents passos explicats anteriorment en l'apartat 6.2.1, combinant els scripts 'ColorSourceManager' i 'ColorSourceView'.

Un cop ja teníem aquests dos elements era el moment de juntar-los de la manera adequada per tal d'aconseguir la màxima sensació de realitat augmentada, de forma que semblés que les tecles efectivament formaven part del món real. Per exemple, en la pròxima imatge podem veure que si directament ajuntem les tecles i la pantalla sense aplicar cap canvi aquesta sensació de realitat augmentada no s'aconsegueix en cap moment.

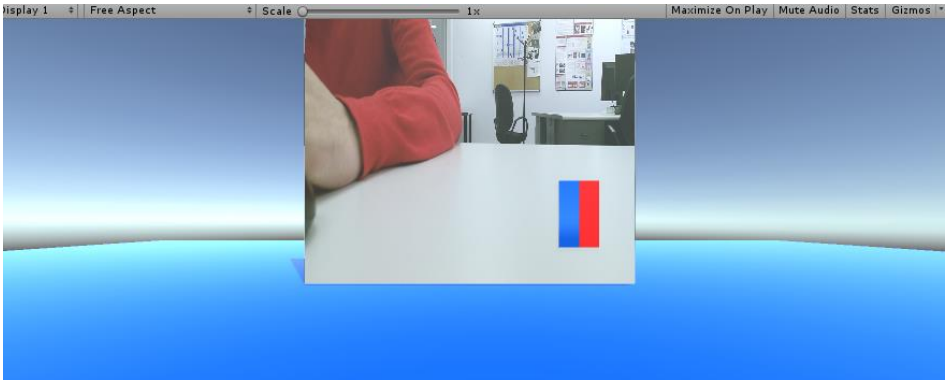



*Figura 20. Tecles introduïdes a la pantalla de manera errònia*

Per aquesta raó, ràpidament es va detectar que les tecles s'havien d'associar dins de la pantalla de manera inclinada per aconseguir profunditat. Després de diversos intents de modificar poc a poc les inclinacions de les tecles es va arribar a les imatges que es mostren a continuació:

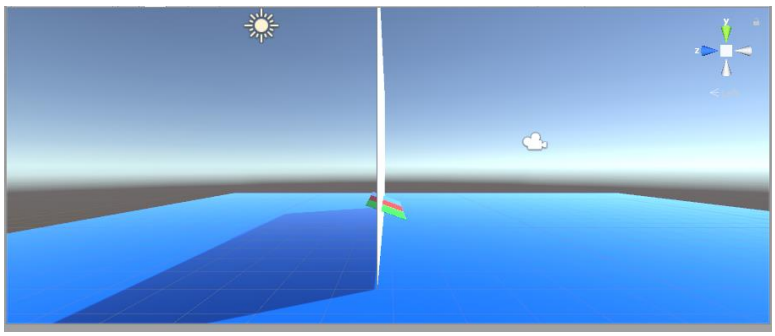


*Figura 21. Tecles incrustades a la pantalla de manera inclinada*

En aquesta primera imatge veiem des d'un punt de vista lateral com s'han incrustat de forma inclinada les tecles com si estiguessin travessant la pantalla.

*Figura 22. Piano situat al terra des del punt de vista de l'usuari*

En aquesta segona imatge ja finalment veiem des del punt de vista de l'usuari el que es mostra per pantalla un cop li donem al 'play' de la nostra escena virtual. Observem com d'aquesta manera les tecles estan perfectament estirades al terra com si fossin objectes reals.

Abans de donar per finalitzada aquesta primera part de l'escena, es va pensar que la posició de les tecles en la pantalla quedaven definides segons l'altura de la Kinect, que estaria recolzada en un trípode. Per això, es va creure convenient crear un codi per tal de que les tecles quedin automàticament posicionades segons l'altura del trípode. A continuació podem veure el codi que s'ha utilitzat:

PosicionTripode.cs | BodyPartStart.cs | SonidoTecla.cs | ColorSourc

Assembly-CSharp

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PosicionTripode : MonoBehaviour {
6
7      public float altura;
8
9      // Use this for initialization
10     void Start () {
11         transform.position += Vector3.up * altura;
12
13     }
14
15     // Update is called once per frame
16     void Update () {
17
18     }
19 }
20
```

*Figura 23. Script Posició Trípode*

En aquest cas es tractava d'un codi simple ja que només calia per una banda crear una variable pública que serà l'altura del trípode i, per altra banda, modificar la posició de la tecla en funció de l'altura. Per aconseguir-ho es va utilitzar les funcions 'transform.position' que s'encarrega de donar la posició actual del Gameobject i finalment 'vector3.up' que és el responsable d'ajustar l'altura.

Un cop afegit el component del script a cada tecla observem en la següent imatge com queda l'inspector d'aquestes, on tenim el script i la variable pública 'altura'.

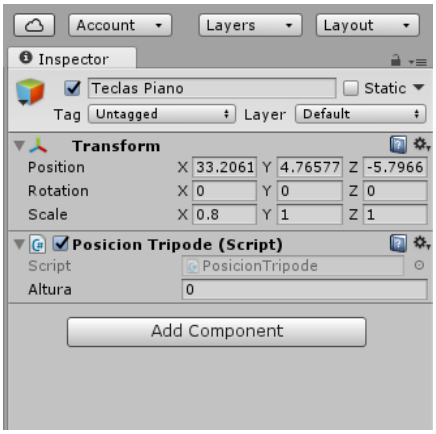



*Figura 24. Elements del GameObject de les tecles*

Arribats a aquest punt havíem assolit la primera part d'aquest projecte, on fins ara podem veure a continuació resumits els diferents elements que formaven part de l'escena: la pantalla, les tecles, el terra i com sempre la llum i la càmera.

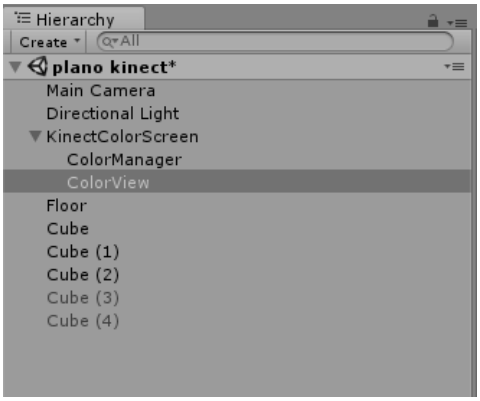



*Figura 25. Finestra de jerarquia de Unity*

## 8.2 Disseny dels joints dels peus i de les tecles de piano falses

En aquesta segona part de l'escena, com s'ha explicat prèviament, es tracta d'aconseguir que quan l'usuari premi les tecles que veu per pantalla es reprodueixi un so diferent per cada tecla.

En primer lloc, per captar els moviments de l'usuari i representar-los hem d'utilitzar els elements que anomenem 'joints' que permeten realitzar el seguiment de tot l'esquelet. Tal i com s'ha comentat anteriorment, de primeres podem pensar que en el nostre model només es necessitaran crear únicament 2 joints que tindran la funció de representar els moviments dels 2 peus de l'usuari. Tanmateix, a causa de la jerarquia que han de seguir tots els joints, sent fills uns d'altres, ha acabat suposant que s'han creat tots els joints d'un esquelet. En cap moment tenir joints extras influenciarà negativament en el projecte, simplement els crearem i només deixarem activats els 2 joints necessaris, tal i com es mostra en la següent imatge:


*Figura 26. Finestra de jerarquia actual de Unity*

Així doncs, per reproduir els moviments de l'usuari es necessitaven 2 elements:

- El primer és el GameObject buit que s'ha explicat a l'apartat 6.2, el BodySourceManager. Tindrà aplicat el comportament a partir del script BodySourcemanager amb que el gestiona la informació de l'esquelet captada per la Kinect.

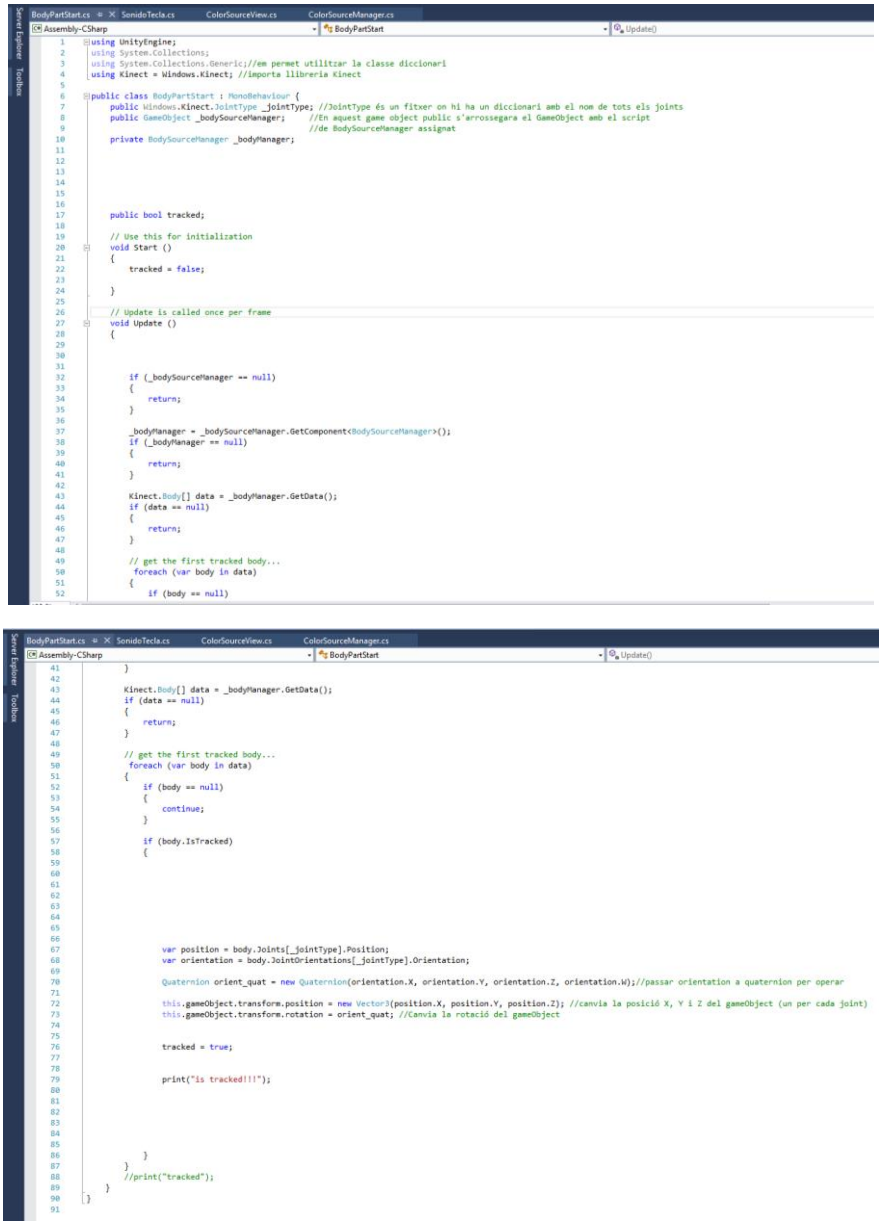
- El segon és un altre GameObject anomenat 'CubeManJoints' que s'encarrega d'emmagatzemar tots els elements necessaris que fan la funció de joints. El comportament d'aquests els gestionarà tots un mateix script que porta el nom de 'BodyPartStart'.

Entendre el funcionament del script 'BodyPartStart és necessari per entendre el funcionament de l'escena. Aquest script té la següent estructura:

1) Es defineixen les variables que s'utilitzaran:
    a) Variable 'joint' tipus JointTypeChild: fitxer on hi ha un diccionari amb el nom de tots els joints
    b) Variable '_bodySourceManager' tipus GameObject.
    c) Variable '_bodyManager'. Variable on s'emmagatzema la informació del seguiment del cos.
    d) Variable 'tracked' de control booleana.

2) Dins la funció Start():
    a) Variable de control booleana pren valor inicial fals

3) Dins la funció Update():
    a) S'introdueixen condicions 'if' per evitar que s'obtinguin dades nul·les i aparegui algun error. En cas que succeeixi, s'aplica un 'return' immediat.
    b) Es detecta el cos més proper
    c) Es guarda la rotació obtinguda de cada joint i se li aplica aquesta rotació a la part del joint pare.



*Figures 27 i 28. Script BodyPartStart*

Un cop implementat el script, cal entendre la declaració de variables d'entrada que aquest codi necessita. En primer lloc, per a cada part del cos (joint) se li assigna un GameObject buit i se li afegeix el component d'aquest script. Un cop fet, a la part de l'inspector (Figura 29) podem observar que tenim l'opció d'introduir 2 variables. A la primera casella de 'Joint Type' només s'ha d'escollir el tipus de joint per tal que l'algoritme obtengui les seves dades i, posteriorment, a la segona casella s'ha d'arrastrar el GameObject que conté el script BodySourceManager.

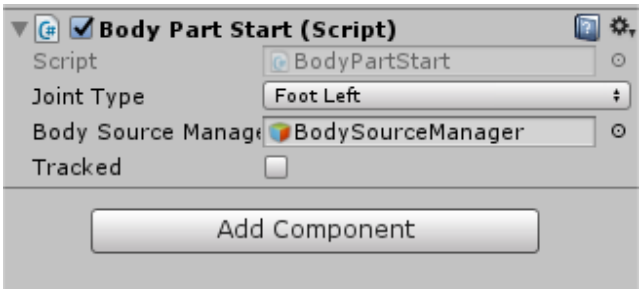



*Figura 29. Opcions del script BodyPartStart*

D'aquesta forma, un cop realitzem aquest procés pels 2 joints necessaris, els 2 peus, i apretem al 'Play' de l'escena visualitzarem 2 cubs que seran la representació dels 2 peus que la Kinect estarà fent el seguiment.

Paral·lelament a la creació dels joints, en aquest apartat de l'escena s'han creat les anomenades 'tecles falses' que consisteixen en les mateixes tecles que es veuen per pantalla i s'han creat en l'apartat anterior. Tanmateix, les anomenem 'falses' perquè aquestes estan apartades de la pantalla juntament amb els 2 joints, o en altres paraules, no són les que l'usuari observarà per prémer. Aquestes seran les que realment reprodueixen quan noten el mínim contacte d'un dels dos joints, que serà quan l'usuari estigui movent el peu per prémer alguna tecla de la pantalla. Per això, aquestes tecles han de tenir 2 components: el primer, un codi associat que els hi permet reproduir un so quan siguin tocades pels joints i, per conseqüència, el segon és el component de so on introduirem la nota que volem que reprodueixi la tecla en qüestió. Pel que fa el codi, en la següent imatge veiem el script que s'ha associat a cada tecla.

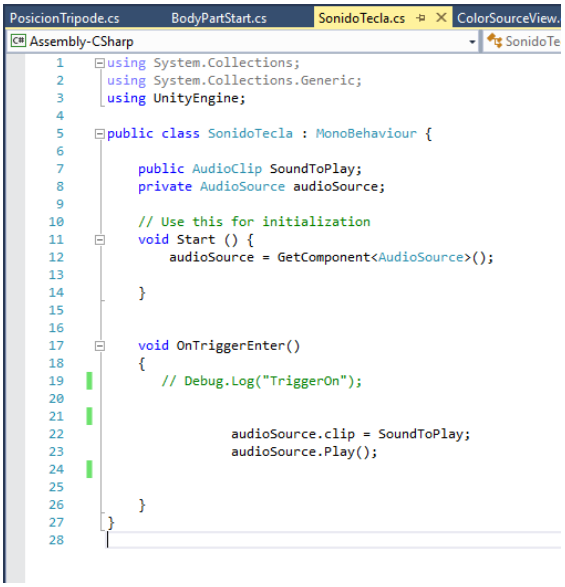



*Figura 30. Script SonidoTecla*

Per aconseguir que la tecla quan sigui apretada reprodueixi la nota musical en qüestió s'ha utilitzat la funció *'OnTriggerEnter'* ja que és la responsable de fer saber al programa quan dos GameObjects xoquen entre ells.

Finalment, podem veure en la següent imatge com queda l'estructura d'una tecla 'falsa' de l'escena, un cop afegides les característiques del so, del script, del color i del *Rigidbody.* Aquesta última és el component que s'aplica als GameObjects quan volem treballar amb moviments físics. En el nostre cas, per tal que l'element es doni compte que l'han apretat i que actuïi en conseqüència necessitem aplicar-li aquesta última condició.

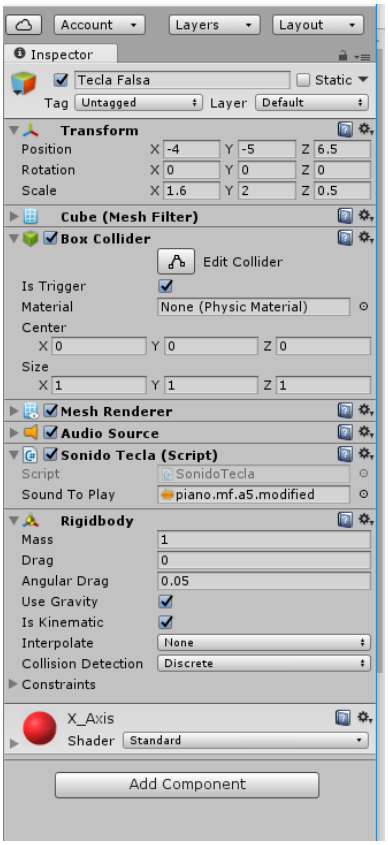



*Figura 31. Inspector d'una tecla falsa*

Les diferents notes musicals que reprodueixen les tecles van ser proporcionades pel meu tutor, les quals podem veure en la següent imatge:

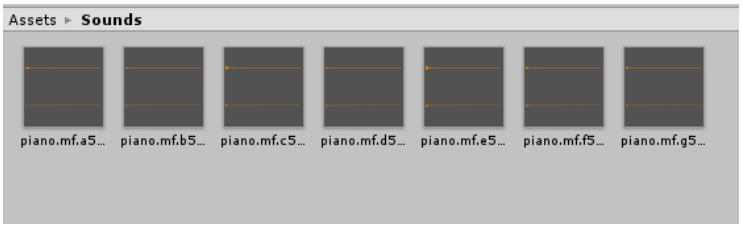



*Figura 32. Arxius de les notes musicals pel piano*

## 8.3 Coordinació d'ambdós dissenys

Després de treballar amb les 2 parts de l'escena només faltava posar-les juntes de manera que no es molestessin entre elles.

Tanmateix, ens podríem preguntar prèviament: Realment era necessari treballar amb dues parts separades o podríem haver-ho posat tot junt? Hem de ser conscients de que la imatge de la càmera és un pla i que per tant no presenta informació sobre la profunditat. Si haguéssim volgut situar els joints dels peus directament sobre les tecles incrustades a la pantalla hauria estat impossible detectar realment quan s'estan prement realment les tecles. Per això, hem hagut de dividir en dues parts l'escena principal, creant unes tecles per la pantalla de l'usuari i unes altres que treballin amb els joints per la seva correcta detecció.

D'aquesta manera, per una banda a la dreta tenim la pantalla on l'usuari es visualitzarà a sí mateix juntament amb les tecles que quedaran incrustades en la pantalla. I per altra banda, tenim els 2 joints que representen els 2 peus de l'usuari juntament amb les 'tecles falses' que no visualitza l'usuari però que són les veritables encarregades de reproduir el so musical. En la següent imatge podem observar com queda distribuït l'espai de l'escena:

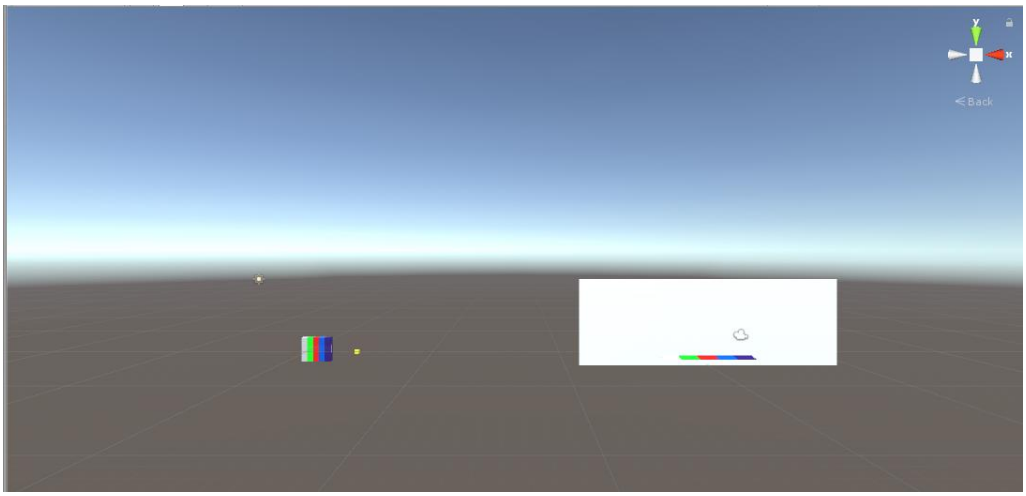



*Figura 33. Escena final*

# 9. Pressupost

Realitzar aquest projecte no ha suposat un gran cost pel que fa el desenvolupament del treball ja que la seva gran majoria es basa en el desenvolupament del software i, per tant, l'impacte econòmic és de petites dimensions. Aquest factor va ser un del punts positius a l'hora de decidir-me per aquest projecte. I és que en el nostre cas tots els dispositius que s'han utilitzat eren de la propietat del CRV i, per conseqüència, l'ús de tots aquests és compartit. Tanmateix, hem de tenir en compte que si s'hagués comprat tots els dispositius únicament per dur a terme aquest projecte sí que estaríem parlant d'un cost elevat.

Per començar, el motor de jocs Unity presenta diverses versions, segons les necessitats de l'usuari. En el nostre cas s'ha treballat amb la versió Personal 5.4.3f1 (64-bit) que destaca per ser simple i es pot adquirir gratuïtament sempre i quan els beneficis no superessin els 100.000$ en el cas que es volgués comercialitzar el projecte. En cap moment ens va preocupar aquest assumpte ja que el nostre projecte és purament acadèmic.

Relacionat encara amb Unity, des de que va sortir la versió 5.0 els usuaris tenen la possibilitat d'importar plugins exteriors a la plataforma sense tenir la versió avançada del programa (versió Pro), valorada en 125$ mensuals.

Seguidament hem de tenir en compte el preu del dispositiu Kinect V2, valorat actualment en 93€ més l'adaptador que presenta un preu de 54€ per tal de poder utilitzar-lo amb l'ordinador. Al mateix temps, per desenvolupar aplicacions amb Kinect cal instal·lar l'SDK de Kinect per Windows. En el nostre cas s'utilitza la versió 2.0 que es pot descarregar gratuïtament des de la web oficial de Microsoft.

En quant a l'ordinador, s'ha de tenir en compte que es requereixen certes especificacions per dur a terme aquest projecte. Es necessita un ordinador amb un sistema operatiu Windows 8 o superior, amb la possibilitat de connexió amb un port USB 3.0, compatibilitat de la targeta gràfica amb Direct X 11 i 4GB de memòria RAM. Concretament, l'ordinador que s'ha utilitzat per aquest projecte és de la propietat del CRV, compleix tots els requisits exposats i està valorat en uns 700€.

L'últim software necessari ha estat el paquet Office amb el que s'ha pogut crear aquesta memòria (Word) i esquemes creats amb PowerPoint. El preu d'una llicència és d'uns 149€.

Finalment, s'han calculat les hores de treball realitzades com enginyer i que han estat valorades en uns 20€/hora. També s'han afegit les hores del professor ponent d'ajuda i revisió del projecte i que han suposat aproximadament 30 hores valorades en 30€/hora.

A continuació es mostra una taula resum (Taula XX) que conté tots els costos totals que s'han exposat:

| | Número d'hores (h) | Cost unitari (€/h) | Cost Total (€) |
|---|---|---|---|
| Unity Personal 5.4.3f1 | - | - | 0€ |
| Kinect v2 | - | - | 93€ |
| Adaptador | - | - | 54€ |
| Ordinador | - | - | 700€ |
| Llicència Office | - | - | 149€ |
| Hores de treball enginyer | 250 | 20 | 5000€ |
| Hores professor ponent | 30 | 30 | 900€ |
| **TOTAL** | - | - | **6896€** |

*Taula 1. Resum del cost total del projecte*

Per tant, podem observar com el cost final del projecte ha estat aproximadament de 6896€.

# CONCLUSIONS

El camp de la realitat virtual es troba actualment en un procés de creixement exponencial i de constant millora. Aquest projecte ha estat una gran oportunitat per tenir una primera idea de les diverses eines que tenim a l'abast quan ens endinsem en aquest món i de l'immens profit que en podem treure. És per això que és molt interessant aprofundir en aquest tema i desenvolupar un projecte relacionat amb la realitat virtual i la sensació d'immersió per part de l'usuari.

Tot això ha estat gràcies a la creació de codis i escenes creades juntament amb la Kinect V2, un dispositiu de gran potencial a l'hora de captar i reproduir els moviments de persones a través dels seus esquelets.

També s'han pogut aprendre i consolidar coneixements sobre programació d'un llenguatge anomenat C#, el qual era totalment desconegut abans de començar aquest projecte i, al mateix temps, aprendre a dominar un motor de videojocs com és Unity.

Pel que fa als objectius plantejats a l'inici del projecte podem afirmar que s'han complert tots amb garanties, aconseguint la sensació d'immersió de l'usuari en l'escena a l'hora de tocar el piano amb els peus.

Per tot això, es pot afirmar que aquest projecte serveix per tenir un primer contacte amb aquest món de la realitat virtual i que pot servir de cara al futur a endinsar-se en camps més complicats i sofisticats.

# Agraïments

En primer lloc voldria agrair al tutor del meu projecte el Toni Susín per donar-me la oportunitat d'introduir-me en aquest món de la realitat virtual que fins ara era totalment desconegut per a mi. Al mateix temps, en moments on no sabia cap a on orientar diverses parts del treball o moments de bloqueig m'ha sabut guiar i ajudar de la millor manera possible.

En segon lloc he d'agrair al Centre de Realitat Virtual de la FME i més concretament al Jordi Moyés per proporcionar-me un espai on treballar còmodament i donar-me totes les eines necessàries, en especial la Kinect V2, per a dur a terme el projecte.

Finalment m'agradaria agrair a la meva família pel suport constant durant tot el projecte.

# Bibliografia

Unity, Tutorial Roll-a-ball

[https://www.youtube.com/watch?v=W_fAidYRGzs, 17 abril 2015]

The Guerrilla CG Project. Euler (gimbal lock) Explained.

[https://www.youtube.com/watch?v=zc8b2Jo7mno, 16 novembre 2016].

David Lopez, Kinect v2 Descripción

[https://es.slideshare.net/DavidLpez60/kinect-v2-descripcin?qid=1d6b37de-d475-44cb-9658-3d31f89f24d7&v=&b=&from_search=1]

The Official Microsoft Blog, La evolución de Kinect y la importancia real de Microsoft Research

[https://www.xatakawindows.com/xbox/la-evolucion-de-kinect-y-la-importancia-de-microsoft-research]

Azwan Jamaluddin, 10 Creative and Innovative Uses of Microsoft Kinect

[https://www.hongkiat.com/blog/innovative-uses-kinect/]